DBMS

Q1. a) What is anamoly in relational model. Explain how normalization can be used to reduce the anamalies

→ • Anamolies in the relational ~~d~~ model are issues that arise due to data redundancy, leading to insertion, deletion anamolies.

• Normalization is the process of organizing a database into well-structured tables by eliminating redundancy and dependency. By applying normal forms, normalization reduces anamolies by:

1. Eliminating redundant data, reducing update anomalies.
2. Organizing data logically, preventing insertion anomalies.
3. Maintaining data integrity, avoiding deletion anamalies.

Q.1.b Explain 2NF and 3NF and BCNF with example
→

- 2NF (Second Normal Form):

A table is in 2NF if:

- It is in 1NF (no repeating ~~found~~ groups).
- All non-key attributes are fully dependent on the entire primary key, not just part of it.

Ex: A table with StudentID, CourseID and CourseName. Since CourseName depends only on CourseID, split it into two tables:

- StudentID, CourseID.
- CourseID, CourseName

- 3NF (Third Normal Form):

A table is in 3NF if:

- It is in 2NF.
- No non-key attribute depends on another non-key attribute.

Ex: A table with StudentID, CourseID, InstructionName.
split it:

- StudentID, CourseID
- CourseID, InstructionName.

- BCNF (Boyce-Codd Normal form):

A stricter 3NF. A table is in BCNF if:

- It is in 3NF.
- Every determinant is a superkey

Ex A table with StudentID, CourseID, InstructorID,
If CourseID isn't unique, split it.

- StudentID, InstructorID.
- InstructorID, CourseID.

Q.2.a) What are relational integrity constraints. Explain with example Domain constraints, Referential integrity and enterprise constraints.

→ Relational Integrity constraints ensure the accuracy & consistency of data in a relational database.

1. Domain constraints:
- These specify the permissible values for a column based on its ~~data~~ data type.

Ex. : A column 'Age' can only have integer values b/w 0 & 120.

2. Referential Integrity:
- Ensures foreign keys correctly reference primary keys in another table.
- Ex: If 'StudentID' in an 'Enrollment' table references 'StudentID' in a 'Students' table, referential integrity ensures you can't enroll a student that doesn't exist.

3. Enterprise Constraints:
- ~~so~~ custom rules defined by the organization to meet specific business requirements.
- Ex.: In a university databases, a constraint that a student can't enroll in more than 5 courses at a time.

Q.2.b) Elaborate the significance of codd's rules. Explain 12 rules proposed by codd's.

→ Codd's 12 rules define the essential properties a database management system must follow to be considered truly relational. They ensure data integrity, consistency & efficient functioning of a relational database.

Rules:

1. Information Rule: Data is stored in tables.
2. ~~Guar~~ Guaranteed Access:
   Each data element can be accessed by table, key & column.
3. Systematic Null Handling: Null values handled consistently
4. Dynamic Catalog: Metadata is accessible like data.
5. Comprehensive Language: supports defining, manipulating, and querying data.
6. View Updatability: Views can be updated.
7. Set-based Operations: Insert, delete, update multiple rows.
8. Physical Independence: Storage changes don't affect data access.
9. Logical Independence: Changes to tables don't affect access.
~~10. Integ~~
10. Integrity Independence: Constraints are stored & enforced separately.
11. Distribution Independence: Data location is transparent to users.
12. Non-subversion: Low-level access doesn't bypass rules

Q.3 a) Explain the concept of conflict serializability with example. .....

→ Conflict Serializability ensures a schedule is equivalent to serial one by a reordering non-conflict operations.

Ex:

Two transactions, T1 & T2, reading & writing X. If T1 writes before T2 reads, reordering ensures the same result as a serial schedule.

- Why emphasize conflict over view serializability?
- Conflict serializability is easier to check & enforce.
- View serializability is harder to test, making conflict serializability more practical for consistency.

Q.B.b Explain the two-phase lock protocol for concurrency control. Also....

→ The Two-Phase Locking (2PL) Protocol ensures serializability ~~for~~ by:

1. Groaing Phase: Acquires locks
2. Shrinking Phase: Release locks.

Strict 2PL:

- Holds write locks until commit/abort.
- Prevents cascading rollbacks.

Rigorous 2PL:

- Holds all locks (read/write) until commit/abort
- Ensures stronger serializability.

- Both controls concurrency and maintain consistency.

(Q.4a) What is R-timestamp(Q) & W-timestamp(Q) Explain the necessary condⁿ used by time stamp ordering protocol to execute for a read / write operation.

→ In the Timestamp Ordering Protocol:

• R-timestamp(Q):
The largest timestamp of any transaction that successfully read data item Q.

• W-timestamp (Q):
The largest timestamp of any transaction that successfully wrote data item Q.

Conditions for Execution:

1. Read (Q):
• If the transaction's timestamp is greater than the W-timestamp (Q), it can read Q.
• Otherwise, the transaction is rolled back.

2. Write (Q):
• If the transaction's timestamp is greater than both R-timestamp (Q) & W-timestamp (Q), it can write to Q.
• Otherwise the transaction is rolled back.

Q.4.b) To ensure atomicity despite failures we use Recovery methods. Explain in detail following Log-Based Recovery ~~meth~~ methods with ex.
i) Deferred Database modifications.
ii) Immediate Database modifications.

→ Log-Based Recovery ensures atomicity by recording all changes in a log before applying them to the database.

i) Deferred Database modifications:
- Changes are not written to ~~be~~ the database until the transaction commits.
- The log keeps track of operations, and after a commit, changes are applied in one go

Ex: Transaction T writes to a log:
1. WRITE (A, 100)
2. Commit. Then, the database is updated

ii) Immediate Database modifications:
- Changes are made immediately ~~a~~ to the database as they occur, even before the transaction commits.

Ex: Transaction T immediately updates:
1. WRITE (A, 100) (log : old value = 50, new value = 100)
2. If failure, undo using log.

(Q.5) a) Compare SQL & NOSQL Database

→

SQL (Relational)	NOSQL (Non-Relational)
1. Structured tabular (rows and columns)	1. Flexible, can be document key-value, graph, or column-family.
2. Fixed schema (predefined)	2. Dynamic schema (flexible structure)
3. Vertically scalable (scale up)	3. Horizontally scalable (scale out)
4. Strong adherence to ACID properties	4. Generally follows CAP theorem, can sacrifice consistency for availability or partition tolerance.
5. Ex.; MySQL, PostgreSQL, Oracle.	5. Mongo DB, Cassandra, Redis, CouchDB

(Q.5.b) Explain BASE Properties of NOSQL Database.

→ BASE properties are used to describe the characteristics of NoSQL databases, focusing on availability and scalability over strict consistency.

- Basically Available:
  The system guarantees availability even if some data is temporarily inconsistent.

- Soft State:
  The state of the system may change over time, even without input, due to eventual consistency.

- Eventual Consistency:
  The system will eventually become consistent after updates are propagated to all nodes.

Q.5.i) Explain Document Based and key value data model of NoSQL Database.
→

→

1. Document - Based Data model:
- Stores data as documents. (e.g : JSON)
- Each document can contain nested structures and key - value pairs.
- Flexible schema allows different documents in the same collection.
-

Ex: {"name" : "John", "age": 30, "address": {"city": "New York", "zip": "10001"}
}

2. Key-Value Data Model:
• Stores data ~~in~~ as key-value pair.

: Each key is unique, retriving its associated value.

· simple and highly scalable, but limited querying capabilities.

Ex: "User 123" : "John Doe".

Q.6.a) Explain CRUD operations used in MongoDB with ex.

→ CRUD stands for Create, Read, Update + Delete which are 4 basic operations.

1. Create:
Used to insert new documents into a collection.
Ex.:
db.users.insertOne({name : "Alice"})

2. Read:
Used to retrieve documents from a collection.
Ex:
db.users.find(); or findOne({name:"Alice"});

3. Update:
Used to modify existing documents in a collection.
Ex: db.users.updateOne({name: "Alice"}, {$set : {age:26}});

4. Delete:
Used to remove documents from a collection

Ex.:
```
db.users.deleteOne({nam: "Alice"});
```

Q.6.b) State & Explain CAP Theorem.

→ The CAP Theorem states that in a distributed data store, you can achieve only two of these properties:

1. Consistency (C):
All nodes have the same data at the same time.

2 Availability (A):
Every request receives a response, even if it's outdated.

3. Partition theorem (P):
The system operates despite network failures.

Trade-offs:

• CP (Consistency and Partition Tolerance):
Consistent data but may sacrifice availability.

• AP (Availability & Partition Tolerance):

Always available but may return stale data.

Q.6.c) Explain Map Reduce with ex.

→ MapReduce is a programming model used for processing large datasets in parallel across a distributed cluster of computers. It consists of two main functions: Map & Reduce.

1. Map Function:
- Takes an input dataset and processes it to produce key-value pairs.
- Ex: If you have a list of words, the Map function counts the occurrence of each word.

Ex: Input: ["apple", "banana"]

Output: ~~F~~ {"apple": 1, "banana": 1}

2. Reduce Function:
- Takes the output from the map function and combines the key-value pairs to produce ~~a~~ final result.

Ex.: The Reduce function sums the counts of each word.

Q.7. a) What are spatial data. Explain Geographic & Geometric data.

→ Spatial data refers to information about the physical location & shape of geometric objects. It is used to represent the positions and relationships of objects in space.

1. Geographic Data:

- Represents real-world locations and is often associated with geographic coordinates.

- Used in mapping applications, Geographic Information systems, and for analyzing spatial relationship.

2. Geometric Data:

- Represents geometric shapes & forms, often defined mathematically.

- Includes points, lines, polygons & 3D shapes that can be used for modeling spatial relationships without direct reference to geographic locations.

Q.7.b) What is the significance of XML databases? Explain with proper ex. when to use XML Database.

Q. 8.b) What is object relational database system? Explain Table inheritence with ex.

→

- An object - Relational Database System combines features of both object - oriented databases & relational databases.
- It allows the storage of complex data types and relationships, enabling more advanced data modeling.
- ORDBMS supports objects, classes, and inheritence while retaining the relational model's benefits.

Table inheritance:

- It allows a child table to inherit properties from a parent table, promoting reuse & organizing data hierarchically.

ex:

1. Parent Table: 'Vehicle'
   - Attributes : vehicle_id, make, model

```
CREATE TABLE vehicle(
        vehicle_id    SERIAL PRIMARY KEY,
        make VARCHAR(50), model VARCHAR(50) );
```

2. Child Table: Car (inherits from vehicle)
   - Additional Attributes: num_doors

```
    CREATE TABLE Car(
            num_doors INT, FOREIGN KEY (vehicle_id) References
            vehicle (vehicle_id)
    ) INHERITS (vehicle);
```